GUNNISON, McKAY & HODGSON, L.L.P.
GARDEN WEST OFFICE PLAZA, SUITE 220
1900 GARDEN ROAD
MONTEREY, CALIFORNIA 93940
(831) 655-0880
FACSIMILE (831) 655-0888

August 25, 2006

Mail Stop Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL LETTER FOR AMENDED APPEAL BRIEF

RE: Applicant(s):	Anatoli Fomenko		
Assignee:	Sun Microsystems, Inc.		
Title:	VERSIONING APPLICATION PROGRAMMING INTERFACE AND METHOD FOR USING VERSIONING FUNCTIONALITY		
Serial No.:	10/081,000	Filed:	February 20, 2002
Examiner:	Etienne P. LeRoux	Group Art Unit:	2161
Docket No.:	P-6507		

Dear Sir:

Transmitted herewith are the following documents in response to the Notice of Non-Compliant Appeal Brief mailed on August 2, 2006 in the above application:

1. Return receipt postcard;
2. This Transmittal Letter (2 pages); and
3. Amended Appeal Brief (53 pages).

Page 14 in the original Appeal Brief has been replaced with a new page 14 in the Amended Appeal Brief to comply with the requirement to state the statue under which each of the claims is rejected.

☒ Conditional Petition for Extension of Time: If an extension of time is required for timely filing of the enclosed documents after all papers filed with this

transmittal have been considered, Applicant(s) hereby petition for such an extension of time.

☒ The Commissioner is hereby authorized to charge any additional fees required for consideration of the enclosed documents, and to credit any overpayment of fees to Deposit Account No. 50-0553.

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on August 25, 2006.

August 25, 2006
Attorney for Applicant(s) Date of Signature

Respectfully submitted,

Forrest Gunnison
Attorney for Applicant(s)
Reg. No. 32,899

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant(s):	Anatoli Fomenko		
Assignee:	Sun Microsystems, Inc.		
Title:	VERSIONING APPLICATION PROGRAMMING INTERFACE AND METHOD FOR USING VERSIONING FUNCTIONALITY		
Serial No.:	10/081,000	Filed:	February 20, 2002
Examiner:	Etienne P. LeRoux	Group Art Unit:	2161
Docket No.:	P-6507		

Monterey, CA
August 25, 2006

Mail Stop Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

AMENDED APPEAL BRIEF

Dear Sir:

Pursuant to 37 CFR § 41.37(a)(1), Appellant files this Appellant's Brief in support of the Notice of Appeal entered by the USPTO on May 10, 2006.

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

REAL PARTY IN INTEREST

The assignee of the above-referenced patent application, Sun Microsystems, Inc., is the real party in interest.

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

RELATED APPEALS AND INTERFERENCES

No other appeals or interferences are known to the undersigned Attorney for Appellant, or the Assignee Appellant, which will directly affect, or be directly affected by, or have a bearing on the Board's decision in this pending Appeal.

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

STATUS OF CLAIMS

Claims 1 to 32 are pending. Claims 1 to 32 stand rejected in the Final Office Action of January 3, 2006. The rejections of Claims 1 to 32 are hereby appealed.

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

STATUS OF AMENDMENTS

In response to the Final Office Action, Appellant filed a paper dated March 3, 2006 requesting reconsideration. An Advisory Action was issued on March 27, 2006, but the entry status of the March 3, 2006 paper was not given in the advisory action.

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

SUMMARY OF CLAIMED SUBJECT MATTER

With respect to Claims 1 to 17, embodiments in accordance with the present invention provide

> . . . a versioning Application Programming Interface (API) 20 for a software platform in accordance with one embodiment of the present invention. The software platform is based on an object-oriented platform-independent programming language, such as Java. The versioning API 20 includes main interfaces 22, a functional implementation 24 of the main interfaces, and a user interface 26 for using the versioning functionality. The main interfaces 22 define versioning functionality and allow access to the versioning functionality. The functional implementation 24 includes classes 42 and libraries 44 implementing the versioning functionality defined by the main interfaces 22. The user interface 26 may include a graphic user interface (GUI) and/or a command line interface (CLI). The versioning API 20 may further include a communication mechanism 28 implementing client-server functionality to perform transactions between remote workspaces. In addition, the versioning API 20 may also include native programming interfaces 48.

Specification, Paragraph [0020].

With respect to claims 18 to 32, embodiments in accordance with the present invention provide:

> . . . a method for using version control functionality via a versioning Application Programming Interface (API) . . . provided in a software platform based on an object-oriented platform-independent programming language. . . . the versioning functionality is defined in main interfaces of the versioning API (200). The defined versioning functionality is implemented in classes and libraries of the versioning API (202). The libraries may include first libraries written in an object-oriented platform-independent programming language, and second libraries written in a native programming language other than the object-oriented platform-independent language. Native programming interfaces are also provided (204). The native programming interfaces (for example, JNIs) allow code written in the object-oriented platform-independent language (for example, Java) to operate with code written in a native language other than the object-oriented

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

> platform-independent language (for example, C and/or C++). The second libraries include native programming interface implementation so as to utilize the native programming interface framework.

Specification, Paragraph [0041].

A summary is provided below for each independent claim and for each dependent claim argued separately.

CLAIM 1

With respect to Claims 1,a versioning application Programming Interface (API) 20 (Fig. 2), (Figs. 6 to 10), {Paragraphs [0020], [0046]} for a software platform based on an object-oriented platform-independent programming language {Paragraph [0020], lines 3, 4} includes main interfaces 22 (Fig. 2) {Paragraphs [0020] lines 4 to 7, [0021]}; 102, 104, 106, 108, 110, 112, 114 (Fig. 3) {Paragraph [0028]}; 200 (Fig. 4) {Paragraph [0041]}.

Main interfaces 22 define versioning functionality. Main interfaces 22 allow access to the versioning functionality{Paragraphs [0020], lines 4 to 7, [0021], [0028], [0041]}.

Versioning API 20 also includes a functional implementation 24 of the main interfaces 22. {Paragraphs [0020], lines 5, and 7 to 9, [0029]}. Functional implementation 24 includes classes 42 (Fig. 2) {Paragraphs [0020], [0021]}; 120-164 (Fig. 3) {Paragraphs [0032] to [0038]} and libraries 44 (Fig. 2) {Paragraph [0038]} implementing the versioning functionality {Paragraph [0020], lines 7 to 9, [0031]}. Classes 42 include a reference to a program module to perform a requested versioning function {Paragraph [0033], lines 2 to 4}.

Versioning API 20 also includes a user interface 26 (Fig. 2) {Paragraphs [0020] lines 6, 9, 10}; 302 (Fig. 5) {Paragraph [0044]} for using the versioning functionality.

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

Note all references to paragraphs in the specification are to the specification as filed and NOT to the Patent Application Publication. The paragraph numbering in the two documents is different.]

CLAIM 2

Claim 2 includes the limitations of Claim 1 as described above. Versioning API 20 also includes a communication mechanism 28 (Fig. 2) implementing client-server functionality {Paragraph [0020] lines 10 to 12}.

CLAIM 3

Claim 3 includes the limitations of Claim 1 as described above. Main interfaces 22 of Claim 1 include an interface {See specification, paragraph [0021] for definition}-defining versioning server functionality 30 {Paragraphs [0028], [0030]}; 104 {Paragraphs [0031], [0032]}; an interface{See specification, paragraph [0020] for definition} defining versioning client functionality 32,{Paragraphs [0028], [0030]}; 102 {Paragraphs [0031] [0032]}; an interface{See specification, paragraph [0020] for definition} defining versioning repository functionality 34 {Paragraph [0028]}; 110 {Paragraphs [0031], [0032]}; an interface defining designated directory structures and access to the designated directory structures 36 {Paragraph [0028]}; 108 {Paragraph [0031]}; and an interface defining transactions between the designated directory structures 38 {Paragraph [0028]}; 112 {Paragraph [0031]}.

CLAIM 4

Claim 3 includes the limitations of Claims 1 and 4 as described above. Main interfaces 22 also include an interface

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

defining file actions within a designated directory structure 40 {Paragraph [0028]}; 114 {Paragraph [0031], [0032]}.

CLAIM 5

Claim 5 includes the limitations of Claim 1 as described above. Versioning API 20 also includes native programming interfaces 48 (Fig. 2) {Paragraphs [0038], [0039], [0040]}, 214 (Fig. 4) {Paragraphs [0042]}, 324 (Fig. 5) {Paragraphs[0045], See also Paragraph [0039]} allowing code written in the object-oriented platform-independent language to operate with code written in a native language other than the object-oriented platform-independent language.

CLAIM 6

Claim 6 includes the limitations of Claims 1 and 5 as described above. Functional implementation 34 includes classes and first libraries written in an object-oriented platform-independent programming language {Paragraphs [0038], [0041]} and second libraries including software routines written in a native programming language other than the object-oriented platform-independent language {Paragraphs [0038], [0041]} The second libraries implement the native programming interfaces 48 (Fig. 2) {Paragraphs [0038], [0039]}, 214 (Fig. 4) {Paragraphs [0042]}, 324 (Fig. 5) {Paragraphs [0045]}.

CLAIM 7

Claim 7 includes the limitations of Claims 1, 5 and 6 as described above. The classes include an implementation class that in turn includes a reference to a first library, said reference being invoked if a requested versioning function is implemented with the object-oriented platform-independent programming language; and a reference to a native function and

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

a second library, said reference being invoked, using a native programming interface, if a requested versioning function is implemented with the native programming language {Paragraph [0039]}.

CLAIM 8

Claim 8 includes the limitations of Claims 1, 5 and 6 as described above. Functional implementation 34 further includes resource files 46 (Fig. 2) {Paragraph [0029]} available to said classes and libraries.

CLAIM 9

Claim 9 includes the limitations of Claim 1 as described above. Classes 42 of versioning API 20 include a class BringoverFrom 140 (Fig. 3) {Paragraphs [0032], [0033], [0034]} including a reference to a program module for copying master files stored in a first directory structure and thereby creating a set of working files; and a class BringoverTo 142 (Fig. 3) {Paragraphs [0032], [0033], [0034]} including a reference to a program module for storing the set of working files in a second directory structure.

CLAIM 10

Claim 10 includes the limitations of Claims 1 and 9 as described above. Classes 42 of versioning API 20 further include a class PutbackFrom 144 (Fig. 3) {Paragraphs [0032], [0033]} including a reference to a program module for copying the working files in the second directory structure and thereby creating a set of updated files; and a class PutbackTo 146 (Fig. 3) {Paragraphs [0032], [0033], [0034]} including a reference to a program module for replacing a corresponding set

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

of the master files in the first directory structure with the set of updated files.

CLAIM 15

Claim 15 includes the limitations of Claims 1, 9, 10 and 14 as described above. Classes 42 of versioning API 20 include a class Lock including a reference to a program module for receive a request for creating a writeable copy of a working file and checking whether a writeable copy of the working file has already been created. {Paragraph [0036]}

CLAIM 16

Claim 16 includes the limitations of Claims 1 and 9 as described above. Classes 42 of versioning API 20 further include a class Freezepoint including a reference to a program module for creating freezepoint files for files in a specified directory structure, the freezepoint files storing a specific time stamp and a then current version of the corresponding files. {Paragraph [0037]}

CLAIM 18

With respect to Claim 18, a method for using version control functionality via a versioning Application Programming Interface (API) 20 (Fig. 2), (Figs. 6 to 10), {Paragraphs [0020], [0046]} is provided in a software platform based on an object oriented platform-independent programming language {Paragraph [0020], lines 3, 4). The method includes defining versioning functionality 200 (Fig. 4) {Paragraph [0041]} in main interfaces 22 (Fig. 2) {Paragraphs [0020] lines 4 to 7, [0021]}; 102, 104, 106, 108, 110, 112, 114 (Fig. 3) {Paragraph [0028]}; 200 (Fig. 4) {Paragraph [0041]} of said versioning API 20; implementing the versioning functionality 200 (Fig. 4)

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

{Paragraph [0041], See also, Paragraphs [0020], lines 4 to 7, [0021], [0028]} in classes 42 (Fig. 2) {Paragraphs [0020], [0021]}; 120-164 (Fig. 3) {Paragraphs [0032] to [0038]} and libraries 44 (Fig. 2) {Paragraph [0038]} of said versioning API, the libraries including: first libraries written in an object-oriented platform-independent programming language {Paragraph [0041]}, and second libraries written in a native programming language other than the object-oriented platform-independent language{Paragraphs [0041], [0042], [0044], [0045]}; and providing native programming interfaces 204 (Fig. 4) {Paragraphs [0041], [0042], [0044], [0045]} allowing code written in the object-oriented platform-independent language to operate with code written in a native language other than the object-oriented platform-independent language, the second libraries including native programming interface implementation.

CLAIM 26

Claim 26 includes the limitations of Claim 18 as described above. The versioning functionality implemented in classes and libraries includes copying master files stored in a first directory structure and thereby creating a set of working files; and storing the set of working files in a second directory structure. {Paragraph [0033]}

CLAIM 27

Claim 27 includes the limitations of Claims 18 and 26 as described above. The versioning functionality implemented in classes and libraries further includes copying the working files in the second directory structure and thereby creating a set of updated files; and replacing the master files in the first directory structure with the set of updated files. {Paragraph [0033]}

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

CLAIM 32

Claim 32 includes the limitations of Claims 18, 26, and 27 as described above. The versioning functionality implemented in classes and libraries also includes creating freezepoint files for files in a specified directory structure, the freezepoint files storing a specific time stamp and a then current version of the corresponding files. {Paragraph [0037]}

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

1. Whether Claims 1, 2, 9 to 14 and 17 are unpatentable under 35 U.S.C. § 102(e) as being anticipated by U.S. Patent Application Publication No. 2002/0116702 of Aptus, hereinafter referred to as Aptus?

2. Whether Claims 3 and 4 are unpatentable under 35 U.S.C. § 102(e) as being anticipated by U.S. Patent Application Publication No. 2002/0116702 of Aptus?

3. Whether Claims 5 to 8 are unpatentable under 35 U.S.C. § 102(e) as being anticipated by U.S. Patent Application Publication No. 2002/0116702 of Aptus?

4. Whether Claims 18 to 31 are unpatentable under 35 U.S.C. § 102(e) as being anticipated by U.S. Patent Application Publication No. 2002/0116702 of Aptus?

5. Whether Claim 15 is unpatentable under 35 U.S.C. § 103(a) as being obvious over U.S. Patent Application Publication No. 2002/0116702 of Aptus, in view of U.S. Patent No. 6,018,743 of Xu, herein after referred to as Xu?

6. Whether Claims 16 and 32 are unpatentable under 35 U.S.C. § 103(a) as being obvious over U.S. Patent Application Publication No. 2002/0116702 of Aptus in view of U.S. Patent No. 6,681,382 of Kakumani, herein after referred to as Kakumani?

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

ARGUMENT

1. Claims 1, 2, 9 to 14, and 17 are patentable over U.S. Patent Application Publication No. 2002/0116702 of Aptus

Claims 1, 2, 9 to 14, and 17 stand rejected under 35 U.S.C. § 102(e) as being anticipated by U.S. Patent Application Publication No. 2002/0116702 of Aptus.

1.A. Requirements for an Anticipation Rejection

To make a prima facie anticipation rejection, the MPEP directs:

> TO ANTICIPATE A CLAIM, THE REFERENCE MUST TEACH EVERY ELEMENT OF THE CLAIM
>
> "A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference.". . . . < "The identical invention must be shown in as complete detail as is contained in the ... claim." *Richardson v. Suzuki Motor Co.*, 868 F.2d 1226, 1236, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989). The elements must be arranged as required by the claim, but this is not an *ipsissimis verbis* test, i.e., identity of terminology is not required.

MPEP § 2131, 8th Ed., Rev. 3, p. 2100-76 (August 2005). It is noted that this directive stated the claim element "**must be**" shown in as complete detail and arranged as required by the claim. Thus, the first issue is what is shown and required by the Claim 1.

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

1.B Claim Interpretation Defines the Elements Set Forth

The MPEP puts forth specific criteria that are to be followed in interpreting a claim. These criteria will be quoted and applied to Claim 1. A comparison of the correct interpretation of Claim 1, as required by the MPEP, with Aptus will demonstrate that the anticipation rejection of Claim 1 is not well founded.

The MPEP requires:

> Office personnel must first determine the scope of a claim by thoroughly analyzing the language of the claim before determining if the claim complies with each statutory requirement for patentability. (Emphasis in original.)

MPEP § 2106 C., 8th Ed., Rev. 3, p 2100-7, 8 (August 2005).

The MPEP further requires:

> Office personnel are to correlate each claim limitation to all portions of the disclosure that describe the claim limitation. This is to be done in all cases, i.e., whether or not the claimed invention is defined using means or step plus function language. The correlation step will ensure that Office personnel correctly interpret each claim limitation.
>
> The subject matter of a properly construed claim is defined by the terms that limit its scope. It is this subject matter that must be examined. (Emphasis added.)

MPEP § 2106 C., 8th Ed., Rev. 3, p 2100-8, (August 2005).

Claim 1 recites in part:

> main interfaces defining versioning functionality, said main interfaces allowing access to the versioning functionality;
>
> a functional implementation of said main interfaces, said functional implementation comprising classes and libraries implementing the versioning functionality, said classes including a reference to a program module to perform a requested versioning function

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

Thus, Claim 1 first recites "interfaces defining version functionality" and characterizes these interfaces as the "main interfaces." According to the above requirements of the MPEP, this element must be correlated to all portions of the disclosure that describe the claim limitation. The above recitation of the summary of Claim 1 is incorporated herein by reference. Appellant also notes that "interfaces" is plural and so represents more than one interface.

Appellant further notes that the correlation of the "interfaces" to the description shows that an explicit definition of interface has been provided in the description, i.e.,

> An interface is a named collection of method definitions and defines a protocol of behavior that can be implemented by any class in the class hierarchy. An interface defines a set of method but does not implement them. An interface may also declare constants.

Specification, Paragraph [0021]. Thus, the "main interfaces" are named collections of method definitions, and have the characteristics provided in this definition.

Claim 1 next recites "a functional implementation of said main interfaces." Thus, based upon the required correlation, the second element of Claim 1 is a functional implementation of the named collection of method definitions in the "main interfaces."

Claim 1 further defines the functional implementation as including classes and libraries. Appellant respectfully notes that these are not just some classes in general, but rather classes in the functional implementation of the main interfaces. Claim 1 provides some further detail with respect to these classes. Correlation of these classes with the specification shows that again a specific definition of these classes is provided, i.e.,

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

> A class that implements the interface agrees to implement all the methods defined in the interface, thereby agreeing to certain behavior. A class is a discrete module of code that has a set of variables (or fields) associated with it, as well as a set of function calls (or methods). Classes that implement an interface inherit the constants defined within that interface. When a class implements an interface, it is essentially signing a contract. (Emphasis Added.)

Specification, Paragraph [0021]. Again, the correlation shows that the recited classes implement specific methods and those methods are those in the main interfaces.

Finally, Claim 1 recites "a user interface for using the versioning functionality." Thus, Claim 1 distinguishes between "main interfaces" that have a functional implementation including classes and a "user interface." This is important, because as described below the rejection confuses the two.

Also, the specification defines user interface as "The user interface 26 may include a graphic user interface (GUI) and/or a command line interface (CLI)" Specification, Paragraph [0020].

Following the MPEP requirement to correlate claim elements with the description shows not only that Claim 1 itself provides specific relationships, but also that the description provides explicit definitions for the elements. As discussed more completely below, the rejection ignores the specification and the teachings in Aptus and uses some generalized definitions that are not supported on the record and at best are simply Examiner argument, which is not sufficient for an anticipation rejection according to the above criteria quoted from the MPEP. In addition, the rejection mixes pieces of Aptus that are described as being at different functional levels--manipulations using a graphic user interface by a user vs. action taken by executing computer code--, which is inappropriate in view of the above quoted requirements from the MPEP with respect to an anticipation rejection.

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

1.C. The Rejection of Claim 1

The final rejection of Claim 1 stated:

> Aptus discloses:
> main interfaces defining versioning functionality, said main interfaces allowing access to the versioning functionality [Fig. 20, 610 paragraph 89]
> a functional implementation of said main interfaces, said functional implementation comprising classes[paragraph 93] and libraries[Fig. 9, paragraph 79] implementing the versioning functionality, said classes including a reference to a program module to perform a requested versioning function[paragraph 93], and
> a user interface for using the versioning functionality[Fig 20, 2002, 2004, 2006, Fig 21, 2102, paragraphs 89, 92 and 93]

Final Rejection, Dated 01/03/2006, pg. 2. The rejection completely ignores the preamble of Claim 1, which gives life, meaning, and vitality to Claim 1.

1.D. The Preamble gives Claim 1 life, meaning, and vitality

Claim 1 recites in part:

> A versioning Application Programming Interface (API) for a software platform based on an object-oriented platform-independent programming language

Thus, the preamble recites that the elements of Claim 1 do not exist in some abstract space, but rather are elements of an application programming interface. Further, the elements of Claim 1 are not part of just some application programming interface, but rather a specific application programming interface. The versioning API of Claim 1 is directed to "a

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

software platform based on object-orientated platform-independent programming language." The ignoring of the preamble and the failure of the rejection to cite any teaching in Aptus that the elements are included in such an API is sufficient to overcome the anticipation rejection by itself because the rejection has failed to establish that Aptus teaches the invention to the same level of detail as recited in Claim 1.

In particular, the MPEP directs:

> "[A] claim preamble has the import that the claim as a whole suggests for it." *Bell Communications Research, Inc. v. Vitalink Communications Corp.*, 55 F.3d 615, 620, 34 USPQ2d 1816, 1820 (Fed. Cir. 1995). **"If the claim preamble, when read in the context of the entire claim, recites limitations of the claim, or, if the claim preamble is 'necessary to give life, meaning, and vitality' to the claim, then the claim preamble should be construed as if in the balance of the claim."** (Emphasis added)

MPEP 2111.02, 8th Ed., Rev. 3, p-2100-51 (August 2005). Thus, the preamble should be construed as if in the balance of the claim, because the elements of Claim 1 take on a particular meaning when so construed. As noted above, Claim 1 first recites an element "main interfaces." Next, Claim 1 characterizes the "main interfaces" by indicating that a functional implementation is required, and then defines the functional implementation further as including "classes and libraries."

A class is an object-oriented programming construct and so the preamble aids in breathing life, meaning, and vitality to this claim element by making the relationship explicit. Similarly, interfaces of an API limit the interfaces to particular types of interfaces, i.e., those found in APIs "for a software platform based on an object-oriented platform-independent programming language." The rejection ignores these facts and rejects the gist of Claim 1 instead of showing that

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

Aptus teaches the elements in the same level of detail as recited in Claim 1 when the elements are considered in view of the preamble. Accordingly, the failure of the rejection to consider the preamble is an admission on the record that Aptus fails to meet the requirements of the MPEP for an anticipation rejection.

1.E. Claim Elements and Aptus Were Improperly construed in the Rejection

As quoted above, the rejection first cited element 610 in Fig. 20 and paragraph [0089] that stated:

> [0089] In addition to the functionality described above, the improved software development tool integrates a version control system that permits programmers using different computers to work simultaneously on a software project by managing the various versions of the source code associated with the software project. The improved software development tool also enables programmers to interact with the version control system by manipulating a diagram or diagram element associated with a software project, thus facilitating the use of the version control system through a more intuitive interface and a more natural grouping of files. For example, FIG. 20 depicts data processing system 2000, which includes a number of computers 2002-2008 connected via a network 2010, where the users of the computers are using the version control system of the improved software development tool 610. On computers 2002-2006, software development tool 610 includes a client component 2012 of the version control system. On computer 2008, the software development tool 610 contains a server component 2014 of the version control system. Computer 2008 is pre-designated as containing a central repository 2016. Central repository 2016 is a shared directory for storing a master copy of project 612. Project 612 comprises all of the source files in a particular software project. Each of the computers 2002-2006 also includes a working directory 2007 that contains working copies of source files that programmers can make changes to without affecting the master copy in the central repository 2016. (Emphasis Added.)

This paragraph describes generally version control system 610. A general description of a version control system that includes a client component, a server component and repositories, as described in paragraph [0089] fails to teach anything about "main interfaces." Nevertheless, in the final rejection, Examiner argument was presented to create such interfaces. See Final Rejection, dated 01/03/2006, pages 10 to 13. At best this argument might be applicable with the right

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

showing in an obviousness rejection because it redefines and characterizes elements in Aptus. However, in anticipation, the MPEP requires the teachings be explicit in the reference as quoted above.

The only use of interface in this paragraph of Aptus is "a more intuitive interface." The intuitive interface being referred to is "The improved software development tool also enables programmers to interact with the version control system by manipulating a diagram or diagram element associated with a software project." The interface is the manipulating a diagram or diagram element associated with the software project by a user, which is manipulation of a graphic user interface and not the "main interfaces," as recited in Claim 1. As noted above, the rejection confuses the elements of Claim 1, i.e., the main interfaces and the user interface. As shown by the citation to Paragraph [0089], the claims limitations have not been interpreted consistently and differentiated.

This is explicitly demonstrated by the next portion of the rejection concerning the functional implementation that cited paragraph [0093] of Aptus. Paragraph [0093] of Aptus stated:

> [0093] An example of a typical user interaction with the version control system via a diagram element will now be described. In this example, it is assumed that the user is viewing a diagram using the improved software development tool and that the diagram visually represents a source file named "Hellojava" that contains a class named "Hello." It is further assumed that the user wishes to verify that he has the most current version of the source code for the "Hello" class by synchronizing his working copy of the file that contains that class with the most current version of the file in the central repository (i.e., he wishes to perform an "Update" command on the file that contains the "Hello" class). With reference to FIG. 21, a user first determines which portion of source code in the software project he wishes to execute a version control command on by visually inspecting diagram 204 of user interface 2102. The user then selects the desired diagram element, in this example diagram element 2104, which corresponds to the class "Hello." The selection is accomplished when the user right clicks within the rectangular area of the diagram element 2104. The selection of diagram element 2104 informs the version control system that the command that will soon be invoked should be performed on the file containing the class "Hello." The user next selects the "Update" command from speedmenu 2202, depicted in FIG. 22, thus providing improved software development tool 610 with an indication of the desired version control command. (Emphasis Added.)

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

Nowhere does this section describe a functional implementation of system 610 that was cited as teaching exactly "main interfaces" in the rejection, as quoted above, but rather a user interaction with a "diagram representing a source file." Again, Aptus is teaching how the user interacts with system 610 via a GUI that presents a diagram representing a source file. This teaches nothing about the classes as that term is used in the claims and teaches nothing about "a reference to a program module to perform a requested versioning function," as recited in Claim 1. Selection of a command using a GUI teaches nothing concerning the underlying program that implements the command.

A class named "hello," as described in this section, does not perform a requested versioning function. Right clicking on a portion of program code in a display does not perform a requested versioning function, but rather according to Aptus "informs the version control system that the command that will soon be invoked should be performed on the file containing the class "Hello."" Invoking a command on a file containing a class is not a class in a functional implementation of main interfaces. Thus, at best, the gist of the invention, rather the invention in the same level of detail as recited in Claim 1, was rejected by citation to paragraph [0093].

Similarly, paragraph [0079] and Fig. 9 do not teach libraries that make up an implementation of the main interfaces as characterized in the rejection, but rather a general process flow diagram for system 610. Thus, the rejection picks a piece from this general description of selecting a template in a template library by system 610 and a part of a user interaction via a graphic user interface from Paragraph [0093] to reject Claim 1.

The teaching of Aptus in the two cited paragraphs is at two fundamentally different levels of abstraction. Paragraph [0079] described what the program does, while paragraph [0093] described user manipulations via a graphic

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

user interface. Appellant notes that the requirement for an anticipation rejection is not whether pieces from different parts of the reference and at two different levels of abstraction and can be extracted and recombined at a single level, but rather the elements in Aptus "must be arranged as required by the claim." This alone also is sufficient to overcome the anticipation rejection of Claim 1. Combining elements from two different levels in Aptus--the program execution associated with selecting a template and user manipulation of a graphic user interface--is not appropriate even in an obviousness rejection.

1.F. An Improper Claim interpretation was used in the Rejection

The Examiner demonstrated that an improper claim interpretation was used in interpreting "interface" as being just some "interface." Specifically,

> Aptus discloses a software development tool with version control which interfaces with programmers using computes 2002-2006 and which also interfaces with multiple versions of the source code being developed for the project. . . Examiner correctly maintains the disclosure of Aptus regarding "interface" in accord with the definition in the Microsoft Computer dictionary.

Final Rejection dated 01/03/2006, pg. 11

Appellant first notes that this statement is an admission that Aptus fails to teach explicitly "main interfaces" and so apparently the Examiner is arguing inherency based on a definition from the Microsoft Computer Dictionary. The resort to a secondary reference demonstrates that the claims were not interpreted as required by the courts and the MPEP.

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

1.F.1 MPEP Tenets with respect to Claim interpretation.

While the Examiner is permitted to interpret the claims broadly, the MPEP and the courts put specific limitations on such an interpretation. In particular, limitations are placed on when a dictionary definition can be used, and in the instant application, such use is inappropriate. Specifically,

CLAIMS MUST BE GIVEN THEIR BROADEST REASONABLE INTERPRETATION

> During patent examination, the pending claims must be "given *>their< broadest reasonable interpretation consistent with the specification." (Emphasis Added.)

MPEP § 2111 8th Ed. Rev. 3, p 2100-46 (August 2005).

The rejection failed to comply with this requirement. The above correlation of the claim elements to the description shows that "the broadest reasonable interpretation consistent with the specification" has been ignored.

Further, the MPEP directs "Office personal **must** rely on the Appellant's disclosure to properly determine the meaning of the claim." (Emphasis added.) Id. This statement alone is sufficient to demonstrate the claim interpretation used in the final rejection is inappropriate. This is not a permissive requirement, but rather is required.

Determining the plain meaning of claim limitations is not reading limitations into the claims as alleged in the final rejection, (Final Rejection, dated 01/03/2006, page 10, "Limitations appearing in the specification but not recited in the claim are not read into the claim.") but rather is complying with the statutory requirements as interpreted by the courts and the PTO. In particular, the MPEP elaborates on the criteria for determining the plain meaning of claim limitations.

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

2111.01 Plain Meaning [R-3]

I. THE WORDS OF A CLAIM MUST BE GIVEN THEIR "PLAIN MEANING" UNLESS THEY ARE DEFINED IN THE SPECIFICATION

> ... During examination, the claims must be interpreted as broadly as their terms reasonably allow. *In re American Academy of Science Tech Center*, **>367 F.3d 1359, 1369, 70 USPQ2d 1827, 1834 (Fed. Cir. 2004). . . . This means that the words of the claim must be given their plain meaning unless applicant has provided a clear definition in the specification. . . . . It is only when the specification provides definitions for terms appearing in the claims that the specification can be used in interpreting claim language. (Emphasis Added.)

MPEP § 2111.01, 8th Ed. Rev. 3, p. 2100-47, 48 (August 2005).

Thus, the MPEP unequivocally describes the limits on using the plain meaning. Appellant has provided a clear definition of at least "interface" and "classes" in the specification as described above and incorporated herein by reference. The above quotes from the MPEP demonstrate that the analysis in the final rejection based on a contrary dictionary definition, or examiner argument as to the definition is not well founded.

The MPEP further unequivocally stated the specification can be used to interpret claim language when a definition is provided as in the instant application. Thus, the MPEP makes clear that such an interpretation is not reading limitations into the claims as stated in the final rejection.

The MPEP further directs:

> ("Where there are several common meanings for a claim term, the patent disclosure serves to point away from the improper meanings and toward the proper meanings.")

MPEP § 2111.01 II., 8th Ed. Rev. 3, p. 2100-49 (August 2005).

Thus, the MPEP directs that when there are several possible meanings, i.e., the one used in the Microsoft Dictionary, the one posited by the Examiner in the rejection, and the one given in the specification, the disclosure serves

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

to point away from the improper meaning, which in this case is the dictionary definition and/or the Examiner's argument used to justify maintaining the anticipation rejection.

A recent CAFC ruling, Philips v AWH Corp, 75 U.S.P.Q.2d 1321, 1334 (CAFC July 2005) reconfirms this point:

> at any time in order to better understand the underlying technology and may also rely on dictionary definitions when construing claim terms, so long as the dictionary definition does not contradict any definition found in or ascertained by a reading of the patent document. (Emphasis Added).

The dictionary definition of "interface" used in the final rejection contradicts the explicit definition given in the specification and so fails to comply with the holding in the case.

Appellant has cited numerous different rulings and they are all consistent on how claim limitations are to be interpreted when an explicit definition is given in the description. The final rejection clearly did not follow these requirements and presented a rationale that directly contradicts the explicit requirements. If there is still any doubt as to how the claim interpretation is to be done when a definition is provided in the specification, the MPEP gives a concise statement:

> Where an explicit definition is provided by the applicant for a term, that definition will control interpretation of the term as it is used in the claim.

MPEP § 2106 C., 8th Ed. Rev. 3, p 2100-8 (August 2005).

Thus, Appellant has demonstrated yet another ground on which the anticipation rejection is not well founded. Only one of these multiple grounds is needed to overcome the anticipation rejection.

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

Finally, the reference relied upon was not made of record during the prosecution and still has not been cited as a proper reference.

1.G. The Microsoft Computer Dictionary Has Not Been Established as a Proper Reference.

With respect to entering a reference in the record, the MPEP directs:

> **707.05(a) Copies of Cited References [R-3]**
>
> Copies of cited >foreign patent documents and non-patent literature< references (except as noted below) are automatically furnished without charge to applicant together with the Office action in which they are cited.

MPEP § 707.05(a), 8th Ed., Rev. 3, p. 700-113 (August 2005).

Even though the Microsoft Computer Dictionary was cited in a footnote and used as justification for the analysis in the final action, a PTO-892 Form was not supplied and a copy of the reference was not supplied to Appellant. Accordingly, the reference was not been made of record during prosecution and should not be allowed to be considered.

To correct this error, the reference and PTO-892 were supplied with the Advisory Action. Even when the reference was supplied in the Advisory Action, the PTO-892 was not properly completed. The MPEP requires:

> For books, minimal information includes the author, title, and date.

MPEP § 707.05(e), 8th Ed., Rev. 3, p. 700-118 (August 2005).

No date is given on the PTO-892 Form supplied with the Advisory Action and so the record fails to establish that the reference is a proper reference with respect to the filing date of the application. Further, the definition cited appears to be incomplete because the last word is "and." This implies

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

that the definition bridges two pages. Unfortunately, this is consistent with the rejection where elements are selectively extracted and the requirements of the MPEP are ignored. The Office should be required to provide a complete citation so that any other information in the definition can be considered and should provide that information on the record. The Office should not be rewarded for not following the requirements of the MPEP by consideration of the reference in this Appeal. Accordingly, while it is unnecessary to reach this point, the reference should not be considered on appeal because the reference was not properly established on the record in a timely fashion.

Claims 2, 9 to 14, and 17 depend from Claim 1 and so distinguish over Aptus for at least the same reasons as Claim 1.

In conclusion, Appellant has explained at multiple levels why Aptus fails to teach the invention to the same level of detail as recited in Claims 1, 2, 9 to 14 and 17. Thus, the Examiner's rejection of Claims 1, 2, 9 to 14 and 17 over Aptus should be reversed.

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

2. Claims 3 and 4 are patentable over Aptus.

Claim 3 recites:

> an interface-defining versioning server functionality;
> an interface defining versioning client functionality;
> an interface defining versioning repository functionality;
> an interface defining designated directory structures and access to the designated directory structures; and
> an interface defining transactions between the designated directory structures

The rejection of Claim 3 cited elements 2014, 2012, and 2016 in Fig. 20 of Aptus against the first three interfaces, respectively, recited in Claim 3; and element 2007 against the fourth and fifth interfaces. (See Final Rejection, dated 01/03/2006, pg. 3)

Aptus taught in Paragraph [0089], as quoted above:

> a server component 2014 of the version control system
> a client component 2012 of the version control system.
> Computer 2008 is pre-designated as containing a central repository 2016
> Each of the computers 2002-2006 also includes a working directory 2007 that contains working copies of source files that programmers can make changes to without affecting the master copy in the central repository 2016

In the rejection of Claim 1, element 610 was cited as teaching exactly main interfaces. Claim 3 further defines the main interfaces. Accordingly, for consistency, the rejection should cite elements that are a part of element 610 in Aptus against Claim 3.

However, in Fig. 20 of Aptus, elements 2016 and 2007 are shown as being separate and distinct from element 610. Thus,

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

the rejection makes up a new structure that is not taught by Aptus. Further, components, a repository, and a working directory of Aptus are all fundamentally different structures from an interface as recited in Claim 3. This is further evidence that the claims were interpreted in a vacuum and not as required by the MPEP.

Each element recited in Claim 3 is an "interface." As noted above, the specification defines:

> An interface is a named collection of method definitions and defines a protocol of behavior that can be implemented by any class in the class hierarchy. An interface defines a set of method but does not implement them. An interface may also declare constants.

Specification, Paragraph [0021]. Thus, the "main interfaces" are named collections of method definitions, and have the characteristics provided in the definition.

As quoted above, the MPEP requires:

> ("Where there are several common meanings for a claim term, the patent disclosure serves to point away from the improper meanings and toward the proper meanings.")

MPEP § 2111.01 II., 8th Ed. Rev. 3, p. 2100-49, 48 (August 2005).

The rejection ignored the definition and the requirements of the MPEP. Instead of citing to a teaching of an "interface is a named collection of method definitions" with the recited protocol of behavior in Claim 3, the rejection relied upon components, a repository and a working directory. Components, a repository and a working directory fail to teach the invention to the same level of detail as recited in Claim 3. Thus, on multiple levels, the rejection of Claim 3 fails to demonstrate that Aptus teaches the invention at the same level of detail as recited in Claim 3. Thus, Claim 3 distinguishes over Aptus for reasons in addition to those with respect to

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

Claim 1, which are incorporated herein by reference. Claim 4 depends from Claim 3 and so distinguishes over Aptus for at least the same reasons as Claims 1 and 3.

In conclusion, Appellant has explained at multiple levels why Aptus fails to teach the invention to the same level of detail as recited in Claims 3 and 4. Thus, the Examiner's rejection of Claims 3 and 4 over Aptus should be reversed.

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

3. Claims 5 to 8 are patentable over Aptus

The rejection of Claim 5 stated:

> Aptus discloses native programming interfaces allowing code written in the object-oriented platform-independent language to operate with code written in a native language [Fig. 2, 202, paragraph 48] other than the object-oriented platform-independent language [Fig. 2, 200, paragraph 48]

Final Rejection, dated 01/03/2006, pg. 3.

This rejection is simply Appellant's claim language with some citations inserted and not any teaching in Aptus. Paragraph [0048] of Aptus taught:

> [0048] As depicted in FIG. 2, source code 202 is being displayed in both a graphical form 204 and a textual form 206. In accordance with methods and systems consistent with the present invention, the improved software development tool generates a transient meta model (TMM) 200 which stores a language-neutral representation of the source code 202. The graphical 204 and textual 206 representations of the source code 202 are generated from the language-neutral representation in the TMM 200. Alternatively, the textual view 206 of the source code may be obtained directly from the source code file. Although modifications made on the displays 204 and 206 may appear to modify the displays 204 and 206, in actuality all modifications are made directly to the source code 202 via an incremental code editor (ICE) 208, and the TMM 200 is used to generate the modifications in both the graphical 204 and the textual 206 views from the modifications to the source code 202.

This section also fails to teach anything concerning an interface that is part of a versioning API. Also, element 202 is source code. Source code 202 is not taught by Aptus as being either part of a versioning API nor native programming interfaces. The MPEP, as quoted above, requires that the reference teach the element recited in the claim. An assignment of an element, i.e., source code 202 to Appellant's

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

claim language is not a teaching in Aptus. Accordingly, the rejection fails to establish a prima facie anticipation rejection.

Further, Claim 5 defines a particular type of interface, a native programming interface. Claim 5 provides a specific definition for the protocol of behavior of such an interface. The comments above with respect to the definition of interface and the requirements of the MPEP with respect to claim interpretation in view of the definition are incorporated herein by reference. The interface allows code written in the object-oriented platform-independent language to operate with code written in a native language. The rejection has failed to cite any teaching of such an interface in source code 202.

Also, the rejection of Claim 5 has failed to show how the source code operates with the transient meta model. In fact, Aptus in paragraph [0048] teaches that the source code is modified with ICE 208 and then the software development tool updates the transient meta model (See paragraph [0080]). Thus, the source code does not operate with the transient meta model because the two are processed sequentially and separately. Thus, Aptus fails to teach the invention to the same level of detail as recited in Claim 5 for multiple different reasons. Thus, Claim 5 distinguishes over Aptus for reasons in addition to those with respect to Claim 1, which are incorporated herein by reference. In addition, each of Claims 6 to 8 depend from Claims 1 and 5 and so distinguish over Aptus for at least the same reasons as these claims.

In conclusion, Appellant has explained at multiple levels why Aptus fails to teach the invention to the same level of detail as recited in Claims 5 to 8. Thus, the Examiner's rejection of Claims 5 to 8 over Aptus should be reversed.

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

4. Claims 18 to 31 distinguish over Aptus.

The final rejection of Claim 18 stated:

> Pâté [Sic] discloses defining versioning functionality in main interdáces [Sic] of said versioning API [fig 20, 610 paragraph 89]; implementing the versioning functionality in classes[paragraph 93] and libraries[Fig. 9, paragraph 79] of said versioning API, the librades [Sic] including first libraries written in an object-oriented platform-independent programming language[Fig. 2, 200, paragraph 48], and second libraries written in a native programming language [Fig. 2, 202, paragraph 48] other than the object-oriented platform-independent language, and providing native programming interfaces allowing code written in the object-oriented platform-independent language to operate with code written in a native language other than the object-oriented platform-independent language, the second libraries including native programming interface implementation[Fig. 2, 204, 206]

Final Rejection, dated 01/03/2006, pg. 6.

The above remarks and quotations from the MPEP and the case law with respect to the requirements for an anticipation rejection, claim interpretation, claim construction, and the preamble are incorporated herein by reference and will not be repeated. Again, the rejection is simply Appellant's claim language with citations inserted.

The rejection completely ignores the preamble of Claim 18. The remarks concerning the preamble with respect to Claim 1 are applicable to Claim 18 and are incorporated herein by reference.

With respect to "main interfaces," "classes" and "libraries" recited in Claim 18, the identical portion of Aptus has been cited as was cited with respect to the "main interfaces," "classes" and "libraries" of Claim 1. The remarks concerning "main interfaces" with respect to Claim 1 are applicable to Claim 18 and are incorporated herein by

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

reference. These remarks alone are sufficient to overcome the anticipation rejections.

Further, as noted above with respect to "classes" and "libraries", the teachings of Aptus in the two cited paragraphs [0079] and [0093] are at two fundamentally different levels of abstraction. Paragraph [0079] described what the program does, while paragraph [0093] described user manipulations of a graphic user interface. Appellant notes that the requirement for an anticipation rejection is not whether pieces from different parts of the reference and at two different levels of abstraction and can be extracted and recombined at a single level, but rather the elements in Aptus " must be arranged as required by the claim." This alone also is sufficient to overcome the anticipation rejection of Claim 18.

However, the rejection of Claim 18 is even further far afield. The libraries taught by Aptus in paragraph [0079] are:

> Such templates are well known in the art. For example, the "Microsoft Foundation Class Library" and the "Microsoft Word Template For Business Use Case Modeling" are examples of standard template libraries from which programmers can choose individual template classes.

Thus, the libraries are template libraries in paragraph [0079] and in the rejection, these libraries are said to teach exactly the libraries of Claim 18. Aptus in paragraph [0079] explicitly explains how templates in the libraries are used, i.e.,

> The software development tool uses the template to parse the source code (step 906), and create the data structure (step 908).

The rejection modifies what is explicitly defined as a template library that provides templates for parsing code and creating a data structure to include two different libraries. No basis for modifying the template library is cited in the final rejection.

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

Element 200 in Fig. 2 is cited as teaching exactly a first library in the template library. As quoted above, Paragraph [0048] of Aptus taught:

> [0048] As depicted in FIG. 2, source code 202 is being displayed in both a graphical form 204 and a textual form 206. In accordance with methods and systems consistent with the present invention, the improved software development tool generates a transient meta model (TMM) 200 which stores a language-neutral representation of the source code 202. The graphical 204 and textual 206 representations of the source code 202 are generated from the language-neutral representation in the TMM 200. Alternatively, the textual view 206 of the source code may be obtained directly from the source code file. Although modifications made on the displays 204 and 206 may appear to modify the displays 204 and 206, in actuality all modifications are made directly to the source code 202 via an incremental code editor (ICE) 208, and the TMM 200 is used to generate the modifications in both the graphical 204 and the textual 206 views from the modifications to the source code 202.

Element 200 is a transient meta model. There is no teaching that this is a library, and there is no teaching that the transient meta model is included in the template libraries that were cited as teaching exactly the "libraries" in Claim 18. In fact, paragraph [0079] teaches that the transient meta model is different from the template library, i.e.,

> After creating the data structure or if there is no existing code, the software development tool awaits an event, i.e., a modification or addition to the source code by the developer (step 910). If an event is received and the event is to close the file (step 912), the file is saved (step 914) and closed (step 916). Otherwise, the software development tool performs the event (step 918), i.e., the tool makes the modification. The software development tool then updates the TMM or model (step 920), as discussed in detail below, and updates both the graphical and the textual views (step 922)

Thus, Aptus describes that the use of the template library is completed and done, and a separate action, "an event" has to occur before anything is even done with the transient meta model. There is absolutely no basis for considering the transient meta model to be a library or that the transient meta model is even part of the template library. Accordingly, the rejection fails to establish that Aptus teaches the invention in Claim 18 to the same level of detail as recited in Claim 18

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

where the first libraries are included in the libraries. This rejection would be inappropriate in an obviousness rejection and cannot possibly be the basis for an anticipation rejection. This alone is sufficient to overcome the anticipation rejection of Claim 18.

The rejection continues simply mischaracterizing Aptus and Claim 18. Claim 18 recites in part:

> libraries of said versioning API, the libraries including:
>
> first libraries written in an object-oriented platform-independent programming language, and
>
> second libraries written in a native programming language other than the object-oriented platform-independent language; and
>
> providing native programming interfaces allowing code written in the object-oriented platform-independent language to operate with code written in a native language other than the object-oriented platform-independent language, the second libraries including native programming interface implementation.

Thus, implementation of the native programming interface is included in the second libraries. However, the rejection cited element 202 as teaching the second libraries and elements 204 and 206 as teaching the native programming interfaces and the implementation of those interfaces. However element 202 is source code, and elements 204 and 206 are displays. One of skill in the art knows that a display is not included in source code 202. The rejection simply has no basis on any level in Aptus or on any level of skill in the art. The rejection mixes parts of Aptus that are not described as libraries and completely ignores the explicit interdependencies defined in Claim 18. Also, depending on the Claim, the interpretation of Aptus changes. Appellant has demonstrated at least four different reasons why the anticipation rejection is not well founded. Only one of these reasons is needed to overcome the rejection.

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

Claims 19 to 31 depend from Claim 18 and so distinguish over Aptus for at least the same reasons as Claim 18.

In conclusion, Appellant has explained at multiple levels why Aptus fails to teach the invention to the same level of detail as recited in Claims 18 to 31. Thus, the Examiner's rejection of Claims 18 to 31 over Aptus should be reversed.

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

5. Claim 15 is patentable over U.S. Patent Application Publication No. 2002/0116702 of Aptus, in view of U.S. Patent No. 6,018,743 of Xu.

Claim 15 stands rejected under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent Application Publication No. 2002/0116702 of Aptus, in view of U.S. Patent No. 6,018,743 of Xu.

Assuming that the combination of the two references is correct and that the interpretation of the two references is correct (Appellant notes that by making these assumptions Appellant does not concede that either of these facts is correct), the additional information from the two references does not correct the deficiencies of the primary reference as noted above with respect to Claim 1 and incorporated herein by reference.

Further, motivation for the combination of references was

> to modify Aptus to include a lock including a reference to a program module for receive a request for creating a writeable copy of a working file and checking whether a writeable copy of the working files has already been created based on the teaching of Xu for the purpose of controlling the updating of the master file such that the integrity of the filed [Sic] is maintained at all times.

Final Rejection, Dated 01/03/2006, pg. 8.

Appellant first notes that the rejection admits that Aptus fails to teach or suggest a class lock. The motivation given for the combination of references shows yet again that Aptus has not been considered as a whole. Aptus taught a way to maintain the integrity of the file at all times and so did not need the modification proposed in the rejection, i.e.,

> If the "Check Out" command were selected (step 2414), the version control system would acquire a copy of one of the versions of a selected file from the central repository, place a copy of the file in the working directory of the requesting computer, and prevent others from checking the file out from the repository (step 2416).

Aptus, Paragraph [0097].

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

Thus, Aptus assured that only one user at a time can make changes to the master copy and so there was no need for a lock, because the system prevented others from checking out the file. In other cases, the user was supplied a read-only copy of the file. Accordingly, one of skill in the art would not be motivated to make the modification and in fact, Aptus teaches away from such a motivation.

In conclusion, Appellant has explained why the combination of Aptus in view of Xu, taken as a whole, does not suggest the subject matter of Claim 15. Thus, the Examiner's rejection of Claim 15 as being unpatentable over the combination of Aptus and Xu should be reversed.

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

6. Claims 16 and 32 are patentable over U.S. Patent Application Publication No. 2002/0116702 of Aptus, in view of U.S. Patent No. 6,681,382 of Kakumani.

Claims 16 and 32 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent Application Publication No. 2002/0116702 of Aptus, in view of U.S. Patent No. 6,681,382 of Kakumani.

Claim 16 depends from Claim 1 and so distinguishes over the combination of references for at least the same reasons as Claim 1 given above, which are incorporated herein by reference. Claim 32 depends from Claim 18 and so distinguishes over the combination of references for at least the same reasons as Claim 18 given above, which are incorporated herein by reference. Thus, the Examiner's rejection of Claims 16 and 32 as being unpatentable over the combination of Aptus and Kakumani should be reversed.

CONCLUSION

For the reasons above, all appealed claims, i.e., Claims 1 to 32, are allowable. Appellant respectfully requests the Board of Patent Appeals and Interferences to reverse the Examiner's various rejections under U.S.C. §§ 102, 103 of these claims.

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

CLAIMS APPENDIX

1. (Original) A versioning Application Programming Interface (API) for a software platform based on an object-oriented platform-independent programming language, said versioning API comprising:

main interfaces defining versioning functionality, said main interfaces allowing access to the versioning functionality;

a functional implementation of said main interfaces, said functional implementation comprising classes and libraries implementing the versioning functionality, said classes including a reference to a program module to perform a requested versioning function; and

a user interface for using the versioning functionality.

2. (Previously Presented) The versioning API according to claim 1, further comprising a communication mechanism implementing client-server functionality.

3. (Previously Presented) The versioning API according to claim 1 wherein said main interfaces comprising:

an interface-defining versioning server functionality;

an interface defining versioning client functionality;

an interface defining versioning repository functionality;

an interface defining designated directory structures and access to the designated directory structures; and

an interface defining transactions between the designated directory structures.

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

4. (Previously Presented) The versioning API according to claim 3 wherein said main interfaces further comprising:

an interface defining file actions within a designated directory structure.

5. (Previously Presented) The versioning API according to claim 1, further comprising:

native programming interfaces allowing code written in the object-oriented platform-independent language to operate with code written in a native language other than the object-oriented platform-independent language.

6. (Previously Presented) The versioning API according to claim 5 wherein said functional implementation comprising:

classes and first libraries written in an object-oriented platform-independent programming language; and

second libraries including software routines written in a native programming language other than the object-oriented platform-independent language, said second libraries implementing said native programming interfaces.

7. (Previously Presented) The versioning API according to claim 6 wherein said classes comprise:

an implementation class including:

a reference to a first library, said reference being invoked if a requested versioning function is implemented with the object-oriented platform-independent programming language; and

a reference to a native function and a second library, said reference being invoked, using a native programming interface, if a requested versioning function is implemented with the native programming language.

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

8. (Previously Presented) The versioning API according to claim 6 wherein said functional implementation further comprising:

resource files available to said classes and libraries.

9. (Previously Presented) The versioning API according to claim 1 wherein the classes comprise:

a class BringoverFrom including a reference to a program module for copying master files stored in a first directory structure and thereby creating a set of working files; and

a class BringoverTo including a reference to a program module for storing the set of working files in a second directory structure.

10. (Previously Presented) The versioning API according to claim 9 wherein the classes further comprise:

a class PutbackFrom including a reference to a program module for copying the working files in the second directory structure and thereby creating a set of updated files; and

a class PutbackTo including a reference to a program module for replacing a corresponding set of the master files in the first directory structure with the set of updated files.

11. (Previously Presented) The versioning API according to claim 10 wherein the classes further comprise:

a class Conflict including:

a reference to a program module for receiving a request for replacing the master files with a set of updated files, and checking for a previous replacement of the master files with another set of updated files;

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

a reference to said class PutbackTo;

a reference to said class BringoverFrom; and a reference to said class BringoverTo.

12. (Previously Presented) The versioning API according to claim 11 wherein said reference to said class PutbackTo is invoked if there is no previous replacement of the master files.

13. (Previously Presented) The versioning API according to claim 11 wherein said reference to said class BringoverFrom and said reference to said class BringoverTo are invoked if there is a previous replacement of the master files.

14. (Previously Presented) The versioning API according to claim 10 wherein the classes further comprise:

a class Checkout including:

a reference to a program module for creating a writeable copy of a working file stored in the second directory structure; and

a reference to a program module for storing the writeable copy to a requested address; and

a class Checkin including:

a reference to a program module for copying the writeable copy of a requested address so as to create an updated working file; and

a reference to a program module for replacing the working file with the updated working file.

15. (Previously Presented) The versioning API according to claim 14 wherein the classes further comprise:

a class Lock including a reference to a program module for receive a request for creating a writeable copy of a working file and checking whether a writeable copy of the working file has already been created.

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

16. (Previously Presented) The versioning API according to claim 9 wherein the classes further comprise:

a class Freezepoint including a reference to a program module for creating freezepoint files for files in a specified directory structure, the freezepoint files storing a specific time stamp and a then current version of the corresponding files.

17. (Previously Presented) The versioning API according to claim 1 wherein said user interface comprises at least one of:

a graphic user interface; and

a command line interface.

18. (Original) A method for using version control functionality via a versioning Application Programming Interface (API) provided in a software platform based on an object oriented platform-independent programming language, said method comprising:

defining versioning functionality in main interfaces of said versioning API;

implementing the versioning functionality in classes and libraries of said versioning API, the libraries including:

first libraries written in an object-oriented platform-independent programming language, and

second libraries written in a native programming language other than the object-oriented platform-independent language; and

providing native programming interfaces allowing code written in the object-oriented platform-independent language to operate with code written in a native language other than the object-oriented platform-independent

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

language, the second libraries including native programming interface implementation.

19. (Previously Presented) The method according to claim 18, further comprising:

receiving, from a client, a request for a versioning function;

calling a class implementing the requested versioning function;

invoking a first library from the class, if the requested versioning function is implemented in the first library written in the object-oriented platform-independent program language; and

using a native programming interface from the class, so as to invoke a second library if a requested versioning function is implemented in the second library written in a native language other than the object-oriented platform-independent language.

20. (Previously Presented) The method according to claim 18 wherein the classes and libraries are mounted with a versioning server application deployed to a hosting server running on the software platform, the software platform being based on the object-oriented platform-independent programming language, said method further comprising:

making a call for a method of a proxy object at the client, the proxy object being associated with a type of versioning transaction;

converting the call for a method to a request of the method;

transmitting the request to the hosting server; and

invoking a servlet at the hosting server to generate a response to the request, the servlet delegating processing of the request to a server object calling a class including the requested method;

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

invoking, from the class, the method directly if the requested method is implemented in a first library written in the object-oriented platform-independent program language; and

invoking, from the class, the method using a native programming interface if the requested method is implemented in a second library written in a native language other than the object-oriented platform-independent program language.

21. (Previously Presented) The method according to claim 20 wherein said making a call is performed with a graphic user interface.

22. (Previously Presented) The method according to claim 20 wherein said making a call is performed with a command line interface.

23. (Previously Presented) The method according to claim 18 wherein said defining versioning functionality comprising:

defining versioning server functionality;

defining versioning client functionality;

defining versioning repository functionality;

defining designated directory structures access to the designated directory structures; and

defining transactions between the designated directory structures.

24. (Previously Presented) The method according to claim 23 wherein said defining versioning functionality further comprising:

defining file actions within a designated directory structure.

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

25. (Previously Presented) The method according to claim 18 wherein in said implementing, the versioning functionality is further implemented in resource files available to the classes and libraries.

26. (Previously Presented) The method according to claim 18 wherein the versioning functionality implemented in classes and libraries comprises:

copying master files stored in a first directory structure and thereby creating a set of working files; and

storing the set of working files in a second directory structure.

27. (Previously Presented) The method according to claim 26 wherein the versioning functionality implemented in classes and libraries further comprises:

copying the working files in the second directory structure and thereby creating a set of updated files; and

replacing the master files in the first directory structure with the set of updated files.

28. (Previously Presented) The method according to claim 27 wherein the versioning functionality implemented in classes and libraries further comprises:

receiving a request for replacing the master files with a set of updated files, and checking for a previous replacement of the master files with another set of updated files;

calling said replacing if there is no previous replacement of the master files since a previous copying of the master files; and

calling said copying master files and said storing if there is a previous replacement of the master files since a previous copying of the master files.

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

29. (Previously Presented) The method according to claim 27 wherein the versioning functionality implemented in classes and libraries further comprises:

creating a writeable copy of a working file stored in the second directory structure; and

storing the writeable copy to a requested address.

30. (Previously Presented) The method according to claim 29 wherein the versioning functionality implemented in classes and libraries further comprises:

copying the writeable copy so as to create an updated working file; and

replacing the working file in the second directory with the updated working file.

31. (Previously Presented) The method according to claim 30 wherein the versioning functionality implemented in classes and libraries further comprises:

receive a request for creating a writeable copy of a working file; and

checking whether a writeable copy of the working file has already been created.

32. (Previously Presented) The method according to claim 27 wherein the versioning functionality implemented in classes and libraries further comprises:

creating freezepoint files for files in a specified directory structure, the freezepoint files storing a specific time stamp and a then current version of the corresponding files.

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

EVIDENCE APPENDIX

None

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

RELATED PROCEEDINGS APPENDIX

None

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on August 25, 2006.

Attorney for Applicant(s)

August 25, 2006
Date of Signature

Respectfully submitted,

Forrest Gunnison
Attorney for Applicant(s)
Reg. No. 32,899
Tel.: (831) 655-0880

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888